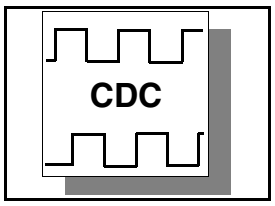
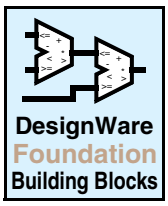




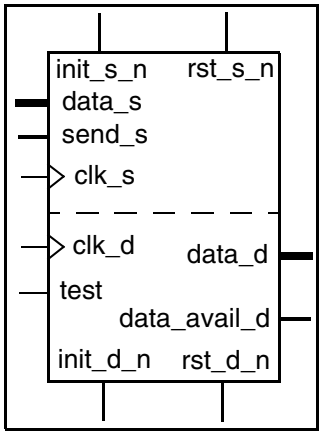
DW_data_qsync_hl

Quasi-Synchronous Data Interface for H-to-L Frequency Clocks

Version, STAR and Download Information: IP Directory

Features and Benefits

- Fully parametrized bus width
- Parameterized clock ratio
- Data available flag in destination clock domain



Description

The DW_data_qsync_hl component implements a 'quasi synchronous' interface between two clock domains where the clocks are related but separate. The interface is parameterizable to allow for a varying bus width and different clock ratios depending on the need in the design.

This synchronizer passes data values from the high frequency domain to the low frequency domain. The two domains are synchronous with respect to each other with the low frequency clock expected to be a derivative of the high frequency clock. Therefore, the derived clock (slow clock) could contain jitter and skew uncertainties with respect to the originating clock source (the fast clock). The *clk_ratio* parameter integer value is determined by the high frequency divided by low frequency (or the slower clock's period divided by the faster clock's period). Full feedback handshake is not used.

Note that when the *clk_ratio* is 2, the *tst_mode* parameter setting is ignored since a negative edge-triggered flip-flop is implemented between the two clock boundaries.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk_s	1	Input	Source Domain clock source
rst_s_n	1	Input	Source Domain asynchronous reset (active low)
init_s_n	1	Input	Source Domain synchronous reset (active low)
send_s	1	Input	Source Domain send request input
data_s	width	Input	Source Domain send data input
clk_d	1	Input	Destination clock source
rst_d_n	1	Input	Destination domain asynchronous reset (active low)
init_d_n	1	Input	Destination domain synchronous reset (active low)
test	1	Input	Scan test mode select
data_d	width	Output	Destination domain data output

Table 1-1 Pin Description (Continued)

Pin Name	Width	Direction	Function
data_avail_d	1	Output	Destination domain data update output

Table 1-2 Parameter Description

Parameter	Values	Description
width	1 to 1024 Default: 8	Vector width of input `data_s` and output `data_d`
clk_ratio	2 to 1024 Default: 2	Integer value of the high speed clock period divided by the low speed clock period
tst_mode	0 to 2 Default: 0	Test mode ■ 0 = No latch is inserted for scan testing ■ 1 = Insert negative-edge capturing register on `data_s` input vector when the `test` input is asserted ■ 2 = Reserved

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

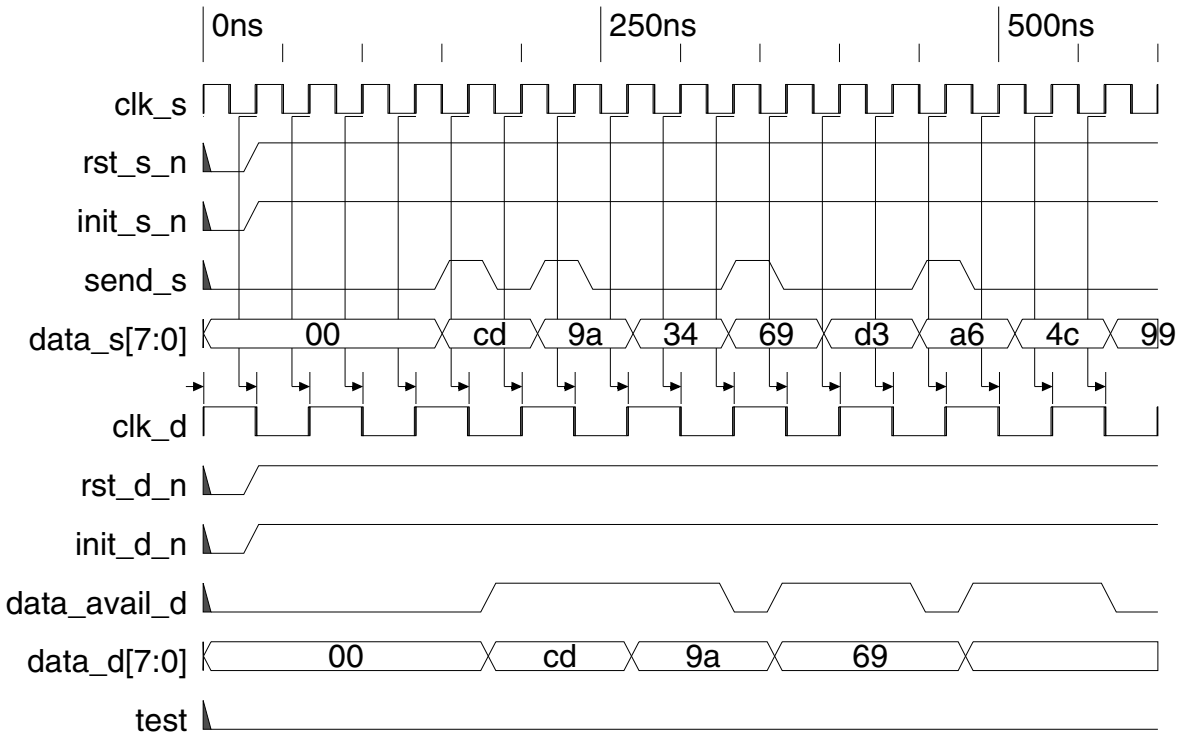
Table 1-4 Simulation Models

Model	Function
DW03.DW_DATA_QSYNC_LH_CFG_SIM	Design unit name for VHDL simulation
dw/dw03/src/DW_data_qsync_lh_sim.vhd	VHDL simulation model source code (modeling RTL)—no missampling
dw/sim_ver/DW_data_qsync_lh.v	Verilog simulation model source code

Timing Diagrams

Figure 1-1 illustrates data transfer between source and destination. Data transferred by the source domain is indicated by the `send_s` pulse. Data '34' and 'd3' on the bus are not transferred.

Figure 1-1 Timing Diagram 1



Related Topics

- Memory – Registers Overview
- DesignWare Building Block IP User Guide

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DW_Foundation_comp.all;

entity DW_data_qsync_hl_inst is
      generic (
        inst_width : NATURAL := 8;
        inst_clk_ratio : NATURAL := 2;
        inst_tst_mode : NATURAL := 0
        );
      port (
        inst_clk_s : in std_logic;
        inst_rst_s_n : in std_logic;
        inst_init_s_n : in std_logic;
        inst_send_s : in std_logic;
        inst_data_s : in std_logic_vector(inst_width-1 downto 0);
        inst_clk_d : in std_logic;
        inst_rst_d_n : in std_logic;
        inst_init_d_n : in std_logic;
        data_avail_d_inst : out std_logic;
        data_d_inst : out std_logic_vector(inst_width-1 downto 0);
        inst_test : in std_logic
        );
    end DW_data_qsync_hl_inst;


architecture inst of DW_data_qsync_hl_inst is

begin

    -- Instance of DW_data_qsync_hl
    U1 : DW_data_qsync_hl
    generic map ( width => inst_width,
                    clk_ratio => inst_clk_ratio,
                    tst_mode => inst_tst_mode )
    port map ( clk_s => inst_clk_s,
                  rst_s_n => inst_rst_s_n,
                  init_s_n => inst_init_s_n,
                  send_s => inst_send_s,
                  data_s => inst_data_s,
                  clk_d => inst_clk_d,
                  rst_d_n => inst_rst_d_n,
                  init_d_n => inst_init_d_n,
                  data_avail_d => data_avail_d_inst,
                  data_d => data_d_inst,
                  test => inst_test );
```

```
end inst;
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_data_qsync_hl_inst( inst_clk_s,
                              inst_rst_s_n,
                              inst_init_s_n,
                              inst_send_s,
                              inst_data_s,

                            inst_clk_d,
                              inst_rst_d_n,
                              inst_init_d_n,
                              data_avail_d_inst,
                              data_d_inst,

                            inst_test );

parameter width = 8;
parameter clk_ratio = 2;
parameter tst_mode = 0;

input inst_clk_s;
input inst_rst_s_n;
input inst_init_s_n;
input inst_send_s;
input [width-1 : 0] inst_data_s;
input inst_clk_d;
input inst_rst_d_n;
input inst_init_d_n;
output data_avail_d_inst;
output [width-1 : 0] data_d_inst;
input inst_test;

    // Instance of DW_data_qsync_hl
    DW_data_qsync_hl #( width,
                        clk_ratio,
                        tst_mode)
      U1 ( .clk_s(inst_clk_s),
           .rst_s_n(inst_rst_s_n),
           .init_s_n(inst_init_s_n),
           .send_s(inst_send_s),
           .data_s(inst_data_s),
           .clk_d(inst_clk_d),
           .rst_d_n(inst_rst_d_n),
           .init_d_n(inst_init_d_n),
           .data_avail_d(data_avail_d_inst),
```

```
                .data_d(data_d_inst),
                .test(inst_test) );

  endmodule
```

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
www.synopsys.com